**DATA FRAME METADATA: ACCESS AND MANIPULATE DATA FRAME DATA SUCH AS COLUMN NAMES, DATA TYPES AND SCHEMA INFORMATION**

|  |  |
| --- | --- |
| 1 | Setup Apache Spark Environment: |
| 2 | Data ingestion |
| 3 | Accessing and Manipulating Metadata |
| 4 | Data Manipulation |
| 5 | Data visualization |

**1.Setup Apache Spark Environment:**

Install Apache Spark on your machine or use a cloud-based Spark service like Databricks.

    Set up a Python environment with necessary libraries like `pyspark`, `matplotlib`, and `pandas`.

**from pyspark.sql import Spark Session**

**import matplotlib. pyplot as plt**

**2.Data Ingestion:**

Download a sample CSV file for this project.

    Start by importing required libraries and creating a Spark Session:

**from pyspark.sql import SparkSession**

**spark = SparkSession.builder \.appName("DataFrame Metadata Project") \**

**.getOrCreate()**

**#Read CSV file into a DataFrame**

**Df = spark.read.csv ('path\_to\_your\_csv\_file.csv', header=True, inferSchema=True)**

**3.Accessing and Manipulating Metadata:**

Explore Data Frame metadata such as column names, data types, and schema information:

**# Show column names**

**print ("Column Names:", df. columns)**

**# Show data types of columns**

**print ("Data Types:", df.dtypes)**

**# Show DataFrame schema**

**df.printSchema()**

**4. Data Manipulation:**

Perform basic data manipulation operations on the DataFrame:

* + **Adding a New Column**
  + **Filtering Data**
  + **Grouping and Aggregating**
  + **Handling Missing Values**
  + **String Manipulation**
  + **Data Type Conversion**

**# Filter DataFrame**

**filtered\_df = df.filter(df['column\_name'] == 'desired\_value')**

**# Group by and aggregate**

**grouped\_df = df.groupBy('column\_name').agg({'numeric\_column': 'mean'})**

**# Join DataFrames**

**joined\_df = df.join(another\_df, on='join\_column', how='inner')**

**# Save DataFrame to a new CSV file**

**joined\_df.write.csv('path\_to\_save\_csv', header=True)**

Data Frame Metadata Operations:

**print Schema ():** This operation prints the schema of a DataFrame, including data types and nullable constraints.

**schema:** Access the schema of a DataFrame programmatically.

**Describe ():** Generates descriptive statistics for numerical and string columns in a DataFrame.

**count ():** Count the number of rows in a DataFrame.

**columns:** Get the list of column names in a DataFrame.

2. Column Metadata Operations:

**metadata:** Access the metadata associated with a column.

**withColumn():** Add or replace metadata for a column.

**drop():** Drop columns from a DataFrame.

**alias():** Rename a column and its associated metadata.

**cast():** Change the data type of a column.

**3. Catalog Metadata Operations:**

**spark.catalog.listDatabases():** List all databases available in the Spark Catalog.

**spark.catalog.listTables(databaseName):** List all tables in a specific database.

**spark.catalog.listColumns(tableName**): List all columns in a table.

**spark.catalog.listFunctions**(); List all available functions in the Spark Catalog.

4. Dataset Metadata Operations (for Parquet, Avro, ORC, etc.):

spark.read.format(format).option("key", "value").load(): Load data with specific options (e.g., schema,metadata).

df.write.format(format).option("key", "value").save(): Write data with specific options (eg, partitioning,

compression, metadata).

5. SQL Metadata Operations:

SHOW DATABASES; Show all databases in the current Spark session.

"SHOW TABLES [IN databaseName); Show all tables in the specified or current database.

DESCRIBE [EXTENDED] tableName; Show detailed information about a table including its metadata.

6. Catalog Functions:

"catalog.getTableMetadata(tableName): Get metadata information for a specific table.

"catalog.createTable(tableName, schema, options): Create a new table with specified metadata and options.

"catalog.dropTable(tableName, ifExists): Drop a table from the catalog.

7. UDF Metadata Operations:

functions.udf(f: AnyRef, dataType: DataType): Register a user-defined function (UDF) with metadata (e.g..

name, description).

**5. Data Visualization:**

    Use matplotlib for data visualization.

Convert DataFrame to Pandas DataFrame for easier plotting:

* matplotlib
* Pandas
* ploty

**import matplotlib. pyplot as plt**

**import pandas as pd**

**# Convert Spark DataFrame to Pandas DataFrame**

**pandas\_df = df.toPandas()**

**# Plotting**

**plt.figure(figsize=(10, 6))**

**plt.scatter(pandas\_df['x\_column'], pandas\_df['y\_column'])**

**plt.xlabel('X Axis')**

**plt.ylabel('Y Axis')**

**plt.title('Scatter Plot')**

**plt.show()**

**accuracy**

**scikitlearn**

**matrix**